Computers and Mathematics with Applications 58 (2009) 1816–1829

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

On the distribution of runs of ones in binary strings

Koushik Sinha [a], Bhabani P. Sinha [b,*]

[a] Honeywell Technology Solutions, 151/1 Bannerghatta Road, Bangalore 560076, India

[b] Indian Statistical Institute, 203 B. T. Road, Calcutta 700108, India

ARTICLE INFO

Article history:
Received 21 July 2008
Received in revised form 6 July 2009
Accepted 14 July 2009

Keywords:
Bernoulli's trial
Run distribution
Counting problem
Generating function,
Run statistics

ABSTRACT

In this paper, we derive the number of binary strings which contain, for a given i_k, exactly i_k runs of 1's of length k in all possible binary strings of length n, $1 \leq k \leq n$. Such a knowledge about the distribution pattern of runs of 1's in binary strings is useful in many engineering applications — for example, data compression, bus encoding techniques to reduce crosstalk in VLSI chip design, computer arithmetic using redundant binary number system and design of energy-efficient communication schemes in wireless sensor networks by transformation of runs of 1's into compressed information patterns, among others. We present, here, a generating function based approach to derive a solution to this counting problem. Our experimental results demonstrate that, for most commonly used file formats, the observed distributions of exactly i_k runs of length k, $1 \leq k \leq n$, closely follow the theoretically derived distributions, for a given n. For $n = 8$, we find that the experimentally obtained values for most file formats agree within $\pm 5\%$ of the theoretically obtained values for all i_k runs of length k, $1 \leq k \leq n$. Also, the root mean square (RMS) values of these deviations across all file types studied in this paper are less than 5% for $n = 8$. In view of these facts, the results presented in this paper could be useful in various application domains, like the ones mentioned above.

1. Introduction

With the advent of the digital computers, binary sequences have assumed great importance in computer science, where they find application in the encoding of data, as well as instructions. Examples of such applications include *run-length encoding* (RLE) of binary strings used in data and image compression, use of redundant binary number system in computer arithmetic, designing energy-efficient data communication schemes in wireless sensor networks by transforming long runs of 1's with suitably encoded messages requiring less energy, and so on.

Run-length encoding finds diverse application in computer science, such as in data and image compression [1]. As an example, the LZ77-based data compression algorithms are a generalization of run-length encoding that can take advantage of runs of strings of characters. Data that have long runs of 1's, such as sound samples, can be run-length compressed after applying a predictive filter such as delta encoding [2,3]. Messom et al. introduced a real-time image processing algorithm [4] based on RLE for a vision based intelligent controller of a Humanoid Robot system. Nagasaka and Miyatake proposed a real-time video scene identification technique [5] that utilizes run length encoding of video feature sequences. Tagaki et al. [6] and De et al. [7] presented high speed VLSI multiplication algorithms that rely on recoding of binary data to redundant number systems to avoid carry propagation and achieve constant time addition by using transformations on runs of 1's.

For HTTP traffic, use of run-length encoding in conjunction with delta encoding can lead to remarkable improvements in response size and response delay for an important subset of HTTP content types [2,3].

* Corresponding author.
E-mail addresses: sinha_kou@yahoo.com (K. Sinha), bhabani@isical.ac.in (B.P. Sinha).

0898-1221/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2009.07.057

Techniques that reduce the amount of data to be transmitted are particularly useful in wireless networks, especially ad hoc and sensor networks, where the devices are highly energy constrained and communication is a major source of energy drain. Energy efficient communication schemes for wireless networks that recode a binary data string into redundant binary by replacing runs of 1's with equivalent codes in the redundant binary number system (RBNS), have been presented in the literature [8–10]. Assuming that each of the 2^n binary strings is equally likely to occur, it was shown by Sinha [8,9] that the fraction of energy saving theoretically obtainable at the transmitter by recoding a binary string of length n in RBNS is, on average, $1 - \frac{n+2}{4n}$ (i.e., about 75% for large n) for noiseless or very low noise channels. This saving reduces to 41% when considering a non-coherent detection based transceiver implementation for an additive white Gaussian noise (AWGN) channel [9]. Their proposed energy efficient communication protocol is called the RBNSiZeComm scheme. However, no energy saving is generated at the receiver by RBNSiZeComm. The *run-zero encoding* (RZE) scheme proposed by Sinha and Sinha [10] is another energy efficient communication protocol that uses recoding of data to RBNS and exploits the statistics on the distribution of runs of 1's (and 0's) in binary strings—similar to the RBNSiZeComm protocol proposed by Sinha [8,9]. The results for the RZE scheme [10] demonstrate that, with a non-coherent detection based receiver, and assuming equal likelihood of all possible binary strings of a given length, there is a 35.2% saving in energy on an average at the transmitter compared to binary FSK, for AWGN channels. Simultaneously, the receiver experiences a saving of 12.5% on average.

In the area of VLSI chip design, as device geometries keep shrinking and clock speeds get faster, coupling capacitance between interconnects has become a major issue. Coupling capacitance occurs when neighboring wires exhibit switching in different directions. The resulting effect, known as bus crosstalk, leads to propagation delay, increase in bus power consumption and even data integrity failure [11–13]. With the current trend of decreasing distance between adjacent bus lines in deep sub micron (DSM) VLSI chip design, there has been a lot of focus in recent times on techniques to reduce bus crosstalk. Most current crosstalk reduction techniques use some bus encoding scheme to map binary patterns with a large number of 0-1 transitions to patterns with fewer such transitions [12–19].

Thus, in many applications like those mentioned above, a detailed knowledge about the distribution of runs in binary strings may be useful in designing efficient algorithms as well as in estimating the benefit of the designed algorithms for a specific application. For example, a prior knowledge about the probability of occurrence of runs of 1's of a given length in a binary string may help us in assessing the merit of a typical run-length encoding scheme. Also, such a knowledge about run statistics may be helpful in estimating the time complexity of multiplication algorithms using redundant binary arithmetic [6]. Similarly, in the area of VLSI chip design, for bus coding schemes to reduce crosstalk, a knowledge of the distribution of runs of 1's (equivalently, runs of 0's) in the data to be transmitted would be very useful, not only in designing effective coding schemes to mitigate crosstalk, but also to analyze the performance of such algorithms.

Distribution of runs in binary strings is closely related to the statistics of success runs in n Bernoulli trials $X_1, X_2, \ldots, X_n$ with success probability p, $0 \leq p \leq 1$ and a failure probability of $q = 1 - p$. The distribution theory of runs has a long history and there is a considerable amount of research work starting from the time around 1940 [20–23] and carried over to recent times [24–27]. A nice review of the theory and application of runs have been done by Balakrishnan and Koutras [28]. The following five important run statistics have been frequently discussed in the literature:

(i) Number of success runs of size exactly k, $1 \leq k \leq n$, in the sense of Mood's counting [21].
(ii) Number of success runs of size greater than or equal to k [26].
(iii) Number of non-overlapping consecutive k successes, in the sense of Feller's counting [29].
(iv) Number of overlapping consecutive k successes, in the sense of Ling's counting [30].
(v) Size of the longest success runs [31].

1.1. Our contribution

To the best of our knowledge, although the above mentioned five important run statistics have been studied frequently in the literature, the problem of a given number (say i_k) of occurrences of consecutive k successes in n Bernoulli trials has not been addressed previously. In the context of binary strings, this problem maps to the occurrence of exactly i_k runs of 1's of length k in a binary string of length n. In the area of energy-efficient communication in wireless sensor networks, this detailed knowledge about the probability of occurrence of runs of 1's of different lengths in binary messages to be transmitted has been successfully used by the RZE scheme [10] as well as by Sinha [8,9] in their RBNSiZeComm protocol to design efficient source encoding mechanisms for reducing the required transmitter (and receiver) energy. Similarly, in VLSI chip designs, such a knowledge of the distribution of runs of 0's and 1's in the data would be very useful for the design and evaluation of crosstalk reduction coding schemes. Currently, crosstalk reduction algorithms typically evaluate the performance of their algorithms using 8-bit to 32-bit randomly generated data [11,13,14,32].

In this paper, we investigate this problem of run distribution and derive an expression for the number of binary strings which contain exactly i_k runs of 1's of length k in all possible binary strings of length n, $1 \leq k \leq n$. We present an elegant generating function based solution to this counting problem.

The derived theoretical results pertain to the situation in which all possible binary strings of a given length n are equally likely to occur. It is thus a natural question as to how closely the real-life data in various applications, where all possible binary strings are not equally likely to occur, follow this run statistics. Such a study will definitely be useful in designing efficient algorithms (e.g., for source encoding of data to be transmitted in energy-efficient communication or

for bus-encoding for reducing the effect of crosstalk) and/or estimating the effectiveness of the devised algorithms in the context of typical real-life applications. For example, several coding schemes for reducing bus crosstalk have been proposed that utilize probability based mapping where the probabilities of 0's and 1's are different [14]. It has been shown that, for instruction address patterns, gray code [16], T0 code [15] and inc-xor [14] perform well to reduce crosstalk while *working zone encoding* [32] can be used for both instruction and data address patterns. For special purpose applications, several techniques have been proposed that attempt to exploit *a priori* knowledge of the characteristics of the patterns to be transmitted [12,15,17,18]. Shin et al. [12] and Zhang et al. [17] have attempted to design bus coding schemes for DSPs and ASICs, and demonstrated the performance of their algorithms on JPEG, MP3 and MPEG files. It has been observed that the behavior of data addresses is different from that of data or instruction addresses, as they are less sequential in nature and typically tend to exhibit some sort of biased statistics in terms of pattern characteristics [12]. Thus, in designing digital signal processor (DSP) and application specific integrated circuits (ASICS), the current research trend is towards exploiting statistical information on the distribution of runs of 0's and 1's in the intended application data to design effective coding schemes that can significantly reduce the number of 0–1 transitions in the patterns to be transmitted and, in the process, reduce crosstalk [12,17–19].

Also, for the energy efficient communication scheme proposed by Sinha [8,9], it was shown that, while the average transmitter energy saving over AWGN channels and using non-coherent detection based receiver is 41% for equal likelihood of all possible binary strings of a given length, considering the data from some real-life applications of wireless sensor networks, the energy savings, however, vary from 33% to 61%.

With this motivation, we have studied the distribution of the occurrence of i_k runs of length k in real-life data of various applications. For this purpose, we have used files of different formats from the EEC test suite proposed by Sinha [8] as it not only covers the range of data types commonly seen in various communication and computation applications in today's world, but it also has a sufficiently large number of samples of each file type to be useful in computing the average distribution of runs of 1's in different file types. The test suite consists of 1100 files of different file types and its description is reproduced here in Section 4 for the convenience of the reader.

The results demonstrate that, for $n = 8$, the experimentally observed values for various file formats agree within $\pm 5\%$ to the theoretically obtained values, except for the plain text files. For plain text files, the deviation from the theoretical values is within $\pm 10\%$, which is explained by the fact that these files contain ASCII characters which have a biased distribution, as all possible binary strings are not equally likely to occur. For $n = 8$, the root mean square (RMS) values of these deviations across all file types is less than 5%, for all i_k runs of length k, $1 \leq k \leq n$. Hence, in view of these observations, we feel that the results presented in this paper could be useful in various application domains such as measuring the benefit of run-length encoding in data compression, reduction in computation time in computer arithmetic with redundant binary number system, design and performance analysis of bus coding schemes in VLSI chips for reducing crosstalk and design of energy efficient communication schemes, among others.

The rest of the paper is organized as follows: Section 2 introduces the notations used in the paper and the basic idea of our approach. A solution to the counting problem using a generating function is presented in Section 3. Section 4 presents the results of counting the distribution of runs of 1's in some popular compression benchmark test suites for a large value of n. Section 5 is the conclusion.

2. Notations and basic ideas

Consider a binary string of length n. We denote a run of 1's of length k occurring in this binary string by R_k. Clearly, two consecutive runs of 1's of length k_1 and k_2, $1 \leq k_1, k_2 \leq n$ in a bit string will be separated by at least one zero. Considering this separating bit occurring on the left of each such R_k, $1 \leq k \leq n$, we denote the symbol $0R_k$ by y_k.

Example 1. Let $n = 8$ and 00111011 be a binary string which contains a run of 1's of length 3 (R_3) and a run of 1's of length 2 (R_2) separated by one zero. This binary string will thus be denoted by $0y_3y_2$.

If the binary string starts with a run of 1's at its leftmost bit position, then we append a zero on the left of the original string to get the symbol y_k for some k, $1 \leq k \leq n$ corresponding to this leftmost run of 1's. We also denote a single zero by the symbol y_0. Then each such y_k, $0 \leq k \leq n$, will be a string of length $k + 1$. We note that, irrespective of whether the given binary string starts with a run of 1's at its leftmost bit position or not, if we always append a zero on the left of the string, then there is a one-to-one correspondence between each such y_k, $1 \leq k \leq n$ in the appended string and a run of 1's in the given binary string. We also note that this appended binary string will be of length $n + 1$.

Example 2. Consider the binary string 11000101 for $n = 8$. We append a zero at the left of the string to get 011000101 of length 9. This zero-appended string will be equivalently denoted by $y_2y_0y_0y_1y_1$. Similarly, appending a zero at the left of the binary string 00111011 of Example 1, we get the string 000111011 of length 9, which will be equivalently denoted by $y_0y_0y_3y_2$. Again, given such a string in terms of the y_k's, $0 \leq k \leq 8$, we can get back the original binary string of length 8 by writing each y_k in terms of 0's and 1's and discarding the leftmost 0 bit.

Let $N_n^{i_k,k}$ denote the total number of binary strings of length n which contain exactly i_k number of R_k's, $1 \leq k \leq n$. Computing $N_n^{i_k,k}$ is equivalent to finding the number of appended binary strings of length $n + 1$ which contains exactly i_k

number of y_k's, $1 \leq k \leq n$. For this counting problem, we proceed in the following way. We first take out the i_k number of y_k's from the appended binary string of length $n+1$ to be left with $n+1-(k+1)i_k$ bits. These $n+1-(k+1)i_k$ bits can be filled with 0's and 1's in all possible ways, but without generating any more y_k. It may be noted that these left over $n+1-(k+1)i_k$ bits may appear in different bit positions which, in general, may not be contiguous. However, from the contiguous bit positions, we can find the y_j's, $j \neq k$. Let us call each such contiguous $j+1$ bits forming a y_j as a block. Thus, the above $n+1-(k+1)i_k$ bits can be partitioned in arbitrary number of blocks, say r blocks, so that each such block will represent some y_j, $0 \leq j \leq n, j \neq k$. Note that this partitioning will be an ordered one, for counting the all possible binary strings.

Example 3. Suppose $n = 8$ and we are looking for the number of binary strings containing exactly 2 runs of 1's of length 1. Examples of such binary strings are 01010111, 01011010, 11100101, and so on. Other than the two appearances of y_1 (=01) in these strings, there are $9-4=5$ bit positions. Let us fill each of the remaining 5 bit positions in the zero-appended strings (of length 9) by a '*'. That is, the zero-appended version of the above three example strings would look like *0101****, *01***01*, and *****0101, respectively. The consecutive appearances of *'s may correspond to some $y_j, j \neq 1$, with each such y_j forming a block of partition of the 5 bit positions holding the * values.

We thus solve the counting problem in the following three steps:
Step 1: (Ordered partitioning of the left over bit positions in a given, say r, number of blocks)
First we take out i_k blocks of size $k+1$ each, $1 \leq k \leq n$, from the integer $n+1$ and then find the total number of ordered partitioning of the integer $t = (n+1)-(k+1)i_k$ in exactly r blocks ($r \geq 1$ for $k < n$ and $r = 0$ for $k = n$), such that there is no block of size $k+1$ in the partition.
Step 2: (Inserting the i_k number of y_k's in between the above partition blocks)
Next, on each such ordered partition obtained in Step 1 above, we insert the i_k blocks of size $(k+1)$ each and find the total number of such distinct possibilities. This step is similar to putting i_k identical balls into $r+1$ boxes, where some boxes may remain empty.
Step 3: (Summing over all possible values of r)
Finally, we sum over the expression obtained in Step 2 above for all possible values of r to get $N_n^{i_k,k}$.
In Section 3 we present a solution to this counting problem using a generating function approach.

3. Generating function based solution

Note that for $k = n$, there is only one such run in the whole string and hence, $N_n^{1,n} = 1$. Using the concept of generating functions [33–35], we use the polynomial $(x + x^2 + x^3 + \cdots)$ to indicate the occurrence of a y_i for some i, $0 \leq i \leq n$ at any bit position in the zero-appended binary string. The purpose of this polynomial is to specify the required length (number of bits) of any y_i at a given position of the string in terms of the power of x of different terms of the polynomial. Thus, the term x corresponds to the occurrence of y_0, x^2 corresponds to y_1, and so on. In general, the occurrence of y_i of length $i+1$ would correspond to the term x^{i+1} of the polynomial. For r blocks of the partition of the integer $t = (n+1)-(k+1)i_k$, each corresponding to some y_i, we have to use the product of r such polynomials. However, to exclude the occurrence of y_k's, we need to subtract x^{k+1} from each of these polynomials. Hence, for r such blocks in the partition of the integer t, we consider the generating function:

$$G(x) = (x + x^2 + x^3 + \cdots - x^{k+1})^r,$$

such that the co-efficient of x^t in $G(x)$ will be the required number of partitions for Step 1.

Now, $G(x)$ can be written as:

$$\begin{aligned}
G(x) &= x^r(1 + x + x^2 + \cdots - x^k)^r \\
&= x^r\Big[\frac{1 - x^k(1-x)}{(1-x)}\Big]^r \\
&= x^r[1 - x^k(1-x)]^r(1-x)^{-r} \\
&= x^r\left[\binom{r}{0} - \binom{r}{1}x^k(1-x) + \binom{r}{2}x^{2k}(1-x)^2 - \cdots + (-1)^q\binom{r}{q}x^{kq}(1-x)^q + \cdots\right] \\
&\quad \times \left[1 + \binom{r}{1}x + \binom{r+1}{2}x^2 + \cdots\right] \\
&= x^r\left[\binom{r}{0} - \binom{r}{1}x^k(1-x) + \binom{r}{2}x^{2k}(1-x)^2 - \cdots + (-1)^q\binom{r}{q}x^{kq}(1-x)^q + \cdots\right] \\
&\quad \times \left[1 + \binom{r}{1}x + \binom{r+1}{2}x^2 + \cdots\right] \\
&= x^r\sum_{q=0}^{r}(-1)^q\binom{r}{q}x^{kq}\sum_{j=0}^{q}(-1)^j\binom{q}{j}x^j\sum_{p=0}^{\infty}\binom{r+p-1}{p}x^p. \qquad (1)
\end{aligned}$$

Table 1
Distribution of $N_n^{i_k,k}$ for $n = 8$.

Runlength size (k)	Values of $N_n^{i_k,k}$			
	$i_k = 4$	$i_k = 3$	$i_k = 2$	$i_k = 1$
1	5	24	66	96
2	–	1	22	97
3	–	–	3	58
4	–	–	–	28
5	–	–	–	12
6	–	–	–	5
7	–	–	–	2
8	–	–	–	1

Table 2
Relative frequency distribution of $N_n^{i_k,k}$ for $n = 8$.

Runlength size (k)	Percentage values of $N_n^{i_k,k}$			
	$i_k = 4$	$i_k = 3$	$i_k = 2$	$i_k = 1$
1	1.9531	9.3750	25.7813	37.5000
2	–	0.3906	8.5938	37.8906
3	–	–	1.1719	22.6563
4	–	–	–	10.9375
5	–	–	–	4.6875
6	–	–	–	1.9531
7	–	–	–	0.7813
8	–	–	–	0.3906

Let $m = n + 1 - (k+1)i_k - r$. Hence, the coefficient of $x^{n+1-(k+1)i_k}$ in $G(x)$ is the coefficient of x^{m+r} in Eq. (1), which is equal to (by setting $p = m - kq - j$),

$$N_m = \sum_{q=0}^{r}\sum_{j=0}^{q}(-1)^{q+j}\binom{r+m-1-kq-j}{m-kq-j}\binom{r}{q}\binom{q}{j}.$$

Hence, the required number $N_n^{i_k,k}$ for $k \geq 1$ is given by,

$$N_n^{i_k,k} = \sum_{r=1}^{n+1-(k+1)i_k}\binom{(r+1)+i_k-1}{i_k}N_m. \tag{2}$$

Expanding the expression for $N_n^{i_k,k}$ from Eq. (2), we obtain the value of $N_n^{i_k,k}$ as,

$$\sum_{r=1}^{n+1-(k+1)i_k}\binom{r+i_k}{i_k}\sum_{q=0}^{r}\sum_{j=0}^{q}(-1)^{q+j}\binom{r+m-1-kq-j}{m-kq-j}\binom{r}{q}\binom{q}{j}. \tag{3}$$

Example 4. For $n = 8$, $k = 2$ and $i_k = 2$, we get the number:

$$\begin{aligned}
N_8^{2,2} &= \sum_{r=1}^{9-6}\binom{r+2}{2}\sum_{q=0}^{r}\sum_{j=0}^{q}(-1)^{q+j}\binom{2-2q-j}{3-r-2q-j}\binom{r}{q}\binom{q}{j}\\
&= \binom{3}{2}\left[\binom{2}{2}\binom{1}{0}\binom{0}{0} - \binom{0}{0}\binom{1}{1}\binom{1}{0} + \binom{-1}{-1}\binom{1}{1}\binom{1}{1}\right] + \binom{4}{2}\left[\binom{2}{1}\binom{2}{0}\binom{0}{0} - 0 + 0\right]\\
&\quad + \binom{5}{2}\left[\binom{2}{0}\binom{3}{0}\binom{0}{0} - 0 + 0\right]\\
&= 0 + 12 + 10 = 22.
\end{aligned}$$

The probability of occurrence of exactly i_k runs of 1's of length k in a given binary string of length n is given by $\frac{N_n^{i_k,k}}{2^n}$. Values of $N_n^{i_k,k}$ for different values of k and i_k and also the probability of occurrence of exactly i_k runs of 1's of length k, when all possible binary strings are equally likely to occur, are shown in Tables 1 and 2, respectively for $n = 8$.

For a given $k \geq 1$, if we now multiply $N_n^{i_k,k}$ by i_k and then sum the product for all possible values of i_k, $1 \leq i_k \leq \lfloor (n+1)/(k+1) \rfloor$, then we would get the total number of occurrences of R_k in all possible strings of length n. Table 3 shows

Table 3
Number of occurrences of runlengths for $n = 8$.

Runlength size (k)	Number of maximum possible values of i_k	Number of occurrences
1	4	320
2	3	144
3	2	64
4	1	28
5	1	12
6	1	5
7	1	2
8	1	1

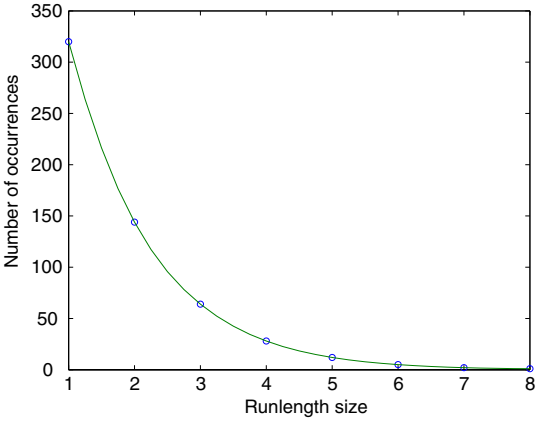

Fig. 1. Number of occurrences of different runlengths for $n = 8$.

the total number of occurrences of all possible run lengths of 1's in all possible binary strings of length 8 bits. Table 3 along with Fig. 1 demonstrates that the number of occurrences of runs of 1's of length k in all possible binary strings of length n, $1 \leq k \leq n$, decreases exponentially as k increases. In fact, this is in complete agreement with the values for the total number of occurrences of runs of 1's of length k as obtained by Sinha [8] which states that, if N_k is the total number of occurrences of runs of 1's of length k in all possible binary strings of length k, then the following recurrence relation holds:

$$\left.\begin{array}{ll} N_{n-k} = 2N_{n-k+1} + 2^{k-2}, & k \geq 2 \\ N_n = 1 & \\ N_{n-1} = 2 & \end{array}\right\}. \tag{4}$$

The solution to the above Eq. (4) is $N_{n-k} = (k+3)2^{k-2}$, for $k \geq 2$.

4. Experimental results

For the purpose of our experiments, we chose the EEC test suite proposed by Sinha [8] as described below, consisting of a good mix of a fairly large number of representative files of different types encountered in real-life applications. We studied the distribution of $N_n^{i_k,k}$, i.e., exactly i_k number of runs of 1's of length k, $1 \leq k \leq n$, for several commonly used file formats, taken from the EEC test suite.

4.1. Test suite description

The *energy efficient communication* (EEC) test suite [8] consists of 1100 files, categorized into 10 different file types. The large number of samples for each representative file type in the test suite makes it very suitable for evaluating the average case distribution of $N_n^{i_k,k}$ in the various file types commonly encountered for computing and communication purposes. Table 4 shows the various file category types and the number of files in each category. Below, we provide a description of each of the categories:

(1) **PDF files**: The pdf files are predominantly research papers and patent documents. Most of them are from the *IEEE*, ACM, *Springer*, *Elsevier* and *MicroPatent* digital libraries.
(2) **Music files**: The music files in the EEC benchmark suite are in MP3, AAC or WAV encoded form and have different bit rates to reflect different audio qualities.

Table 4
Energy efficient communication (EEC) test suite.

	File type	Number of files
1	Postscript documents	60
2	Adobe Acrobat documents	78
3	Music files	107
4	HTML files	71
5	Image files	71
6	Binaries	82
7	MS documents	112
8	Plain text files	122
9	Video files	52
10	Streaming video	345

Table 5
Relative frequency distribution of $N_n^{i_k,k}$ in binary files (%) with $n = 8$.

Runlength size (k)	Percentage values of $N_n^{i_k,k}$			
	$i_k = 4$	$i_k = 3$	$i_k = 2$	$i_k = 1$
1	1.0051	7.0426	25.5013	40.9192
2	–	0.0666	5.7280	35.7592
3	–	–	0.3806	18.3461
4	–	–	–	6.2420
5	–	–	–	2.2752
6	–	–	–	1.5204
7	–	–	–	0.2058
8	–	–	–	0.3695

(3) **BMP files**: This data set is a collection of frames in (bitmap format) of real-time streaming video obtained from a surveillance camera.
(4) **Plain text files**: These files represent ASCII data and are a mix of log files, configuration files and program source codes written in different languages such as C, C++ and Java.
(5) **HTML files**: The HTML files in the suite reflect the common activity patterns of users on the internet in today's world, such as surfing, news reading, searching and internet shopping. They consist of pages from sites such as Yahoo, MSN, Amazon, etc., pages from the popular web based email sites, search results from Google, Yahoo, MSN and other search engines and pages from auction sites such as eBay.
(6) **Image files**: These are a mix of indoor and outdoor pictures in JPEG format with a higher percentage of outdoor pictures.
(7) **Video files**: These are either MPEG, WMV or ASX encoded files and have different bit rates to reflect different video qualities.
(8) **Binary files**: The binary files are a mix of Microsoft Windows dynamic link library (DLL) files, object files, Linux and Microsoft's executable binaries, Java *class* files and Linux and Microsoft's library files.
(9) **MS Document files**: The files in this category reflect the various commonly used document classes from Microsoft such Word documents, PowerPoint presentations and Excel spreadsheets.

4.2. Results details

For each of the chosen file types in our experiment, we used 50 representative files from the EEC test suite to derive the relative frequency distribution of $N_n^{i_k,k}$ values for each file type. The relative frequencies corresponding to each $N_n^{i_k,k}$ for a particular file type was obtained by dividing the total occurrences of exactly i_k runs of length k by the total number of frames required to read the entire set of files. Tables 5–10 present the results of our experimentation on different file types for $n = 8$ as percentage values. All simulations were done using C and MATLAB.

Deviations in percentage values of $N_n^{i_k,k}$ from the theoretically calculated values for $n = 8$ have been shown in files of different types, such as binary files, plain text files, JPEG image files, MPEG video files, MP3 music files and PDF documents in Tables 11–16, respectively. From these tables, it appears that the deviation in percentage values of $N_n^{i_k,k}$ for different k and i_k from the theoretically calculated values always lies within $\pm 5\%$ for all file types excepting the plain text files, for which this deviation rises up to nearly $\pm 10\%$. An apparent explanation for this larger deviation in case of plain text files may be given as follows: plain text files consist of ASCII characters which require a byte of memory to store and, hence, the runs of 1's in such files are constrained to have some biased distribution (i.e., all possible bit patterns are not equally likely). In contrast, the other file formats do not have such limitations on the size of the run-lengths, and their behavior in the context of such run statistics is somewhat closer to the situation where all possible binary strings are equally likely to occur.

The *root mean square* (RMS) *deviations* in percentage values from the theoretically obtained ones over all such studied file types for different k and i_k, have also been tabulated in Table 17 for $n = 8$. From the results in Table 17 we see that the largest

Table 6
Relative frequency distribution of $N_n^{i_k,k}$ in plain text files with $n = 8$.

Runlength size (k)	Percentage values of $N_n^{i_k,k}$			
	$i_k = 4$	$i_k = 3$	$i_k = 2$	$i_k = 1$
1	0.3011	5.3869	26.1517	46.6649
2	–	0.0000	7.2936	41.6686
3	–	–	0.5089	19.9026
4	–	–	–	6.2709
5	–	–	–	1.6242
6	–	–	–	0.0159
7	–	–	–	0.0000
8	–	–	–	0.0000

Table 7
Relative frequency distribution of $N_n^{i_k,k}$ in JPEG image files with $n = 8$.

Runlength size (k)	Percentage values of $N_n^{i_k,k}$			
	$i_k = 4$	$i_k = 3$	$i_k = 2$	$i_k = 1$
1	2.2047	9.3796	25.1922	37.2033
2	–	0.4427	8.9378	38.1596
3	–	–	1.4165	24.7618
4	–	–	–	12.3270
5	–	–	–	4.8993
6	–	–	–	2.0500
7	–	–	–	0.8248
8	–	–	–	0.3721

Table 8
Relative frequency distribution of $N_n^{i_k,k}$ in MPEG video files with $n = 8$.

Runlength size (k)	Percentage values of $N_n^{i_k,k}$			
	$i_k = 4$	$i_k = 3$	$i_k = 2$	$i_k = 1$
1	1.9810	9.7533	26.2008	37.5341
2	–	0.3235	7.5418	36.7832
3	–	–	0.9839	20.9681
4	–	–	–	9.3665
5	–	–	–	3.9803
6	–	–	–	1.6952
7	–	–	–	0.7205
8	–	–	–	0.7970

Table 9
Relative frequency distribution of $N_n^{i_k,k}$ in MP3 music files with $n = 8$.

Runlength size (k)	Percentage values of $N_n^{i_k,k}$			
	$i_k = 4$	$i_k = 3$	$i_k = 2$	$i_k = 1$
1	2.1643	10.4993	27.2170	36.8658
2	–	0.3561	8.2685	38.0352
3	–	–	0.9879	20.5504
4	–	–	–	9.1809
5	–	–	–	3.9057
6	–	–	–	1.5626
7	–	–	–	0.6573
8	–	–	–	0.8944

RMS deviation (4.104208%) occurs for $k = 1$, $i_k = 1$. Given that the RMS deviations for all i_k and k, $1 \leq k \leq n$, is less than 5% across all the studied file types, we can hence conclude that the distribution of $N_8^{i_k,k}$ in most of the popular file formats closely resembles the theoretical distribution. However, it is difficult to give any theoretical bound on such deviation unless the files can be properly characterized with the probability of occurrences of the different binary strings in those respective file types.

For $n = 64$, we have also plotted both the theoretical as well as the experimentally obtained distribution of $N_n^{i_k,k}$ for different file types, with $k = 1$–6. For the sake of comparison, in each of these cases which we have studied, the two curves showing the experimentally obtained values and the theoretical values have been plotted together, as shown in Figs. 2–7. While the dotted curves represent the theoretical values, the solid lines correspond to the experimentally observed values.

Table 10
Relative frequency distribution of $N_n^{i_k,k}$ in PDF documents with $n = 8$.

Runlength size (k)	Percentage values of $N_n^{i_k,k}$			
	$i_k = 4$	$i_k = 3$	$i_k = 2$	$i_k = 1$
1	1.5193	7.9493	23.9422	39.7112
2	–	0.3413	8.5737	39.2751
3	–	–	0.9149	21.3009
4	–	–	–	10.5156
5	–	–	–	4.2282
6	–	–	–	1.8359
7	–	–	–	0.8546
8	–	–	–	0.9977

Table 11
Deviation in percentage values of $N_n^{i_k,k}$ in binary files from the theoretical values with $n = 8$.

Runlength size (k)	Deviation in percentage values of $N_n^{i_k,k}$			
	$i_k = 4$	$i_k = 3$	$i_k = 2$	$i_k = 1$
1	−0.9480	−2.3324	−0.2800	3.4192
2	–	−0.3240	−2.8658	−2.1314
3	–	–	−0.7913	−4.3102
4	–	–	–	−4.6955
5	–	–	–	−2.4123
6	–	–	–	−0.4327
7	–	–	–	−0.5755
8	–	–	–	−0.0211

Table 12
Deviation in percentage values of $N_n^{i_k,k}$ in plain text files from the theoretical values with $n = 8$.

Runlength size (k)	Deviation in percentage values of $N_n^{i_k,k}$			
	$i_k = 4$	$i_k = 3$	$i_k = 2$	$i_k = 1$
1	−1.6520	−3.9881	0.3704	9.1649
2	–	−0.3906	−1.3002	3.7780
3	–	–	−0.6630	−2.7537
4	–	–	–	−4.6666
5	–	–	–	−3.0633
6	–	–	–	−1.9372
7	–	–	–	−0.7813
8	–	–	–	−0.3906

Table 13
Deviation in percentage values of $N_n^{i_k,k}$ in JPEG image files from the theoretical values with $n = 8$.

Runlength size (k)	Deviation in percentage values of $N_n^{i_k,k}$			
	$i_k = 4$	$i_k = 3$	$i_k = 2$	$i_k = 1$
1	0.2516	0.0046	−0.5891	−0.2967
2	–	0.0521	0.3440	0.2690
3	–	–	0.2446	2.1055
4	–	–	–	1.3895
5	–	–	–	0.2118
6	–	–	–	0.0969
7	–	–	–	0.0435
8	–	–	–	−0.0185

Figs. 2–4 provide a comparison between the theoretical values and the distribution of $N_{64}^{i_k,k}$ in binary files for $k = 1$–6. Fig. 5 depicts the distribution of $N_{64}^{i_k,k}$ in MP3 and MPEG video files and Fig. 6 shows the corresponding distribution in PDF and text files for $k = 4$. Fig. 7 shows the distribution of $N_{64}^{i_k,k}$ in JPEG and Microsoft document files for $k = 2$. From all these figures, it appears that the distribution of $N_n^{i_k,k}$ in most of the popular file formats found in various applications closely follows the theoretical distribution. For plain text files, these experiments with many other values of n and k reveal that the deviations in the experimentally obtained distributions of $N_n^{i_k,k}$ from the theoretical ones are small with large values of k (as shown in Fig. 6(b) with $n = 64$ and $k = 4$), while these deviations are more with smaller values of k (as shown in Fig. 8 with $n = 64$

Table 14
Deviation in percentage values of $N_n^{i_k,k}$ in MPEG video files from the theoretical values with $n = 8$.

Runlength size (k)	Deviation in percentage values of $N_n^{i_k,k}$			
	$i_k = 4$	$i_k = 3$	$i_k = 2$	$i_k = 1$
1	0.0279	0.3783	0.4195	0.0341
2	–	−0.0671	−1.0520	−1.1074
3	–	–	−0.1880	−1.6882
4	–	–	–	−1.5710
5	–	–	–	−0.7072
6	–	–	–	−0.2579
7	–	–	–	−0.0608
8	–	–	–	−0.4064

Table 15
Deviation in percentage values of $N_n^{i_k,k}$ in MP3 music files from the theoretical values with $n = 8$.

Runlength size (k)	Deviation in percentage values of $N_n^{i_k,k}$			
	$i_k = 4$	$i_k = 3$	$i_k = 2$	$i_k = 1$
1	0.2112	1.1243	1.4357	−0.6342
2	–	−0.0345	−0.3253	0.1446
3	–	–	−0.1840	−2.1059
4	–	–	–	−1.7566
5	–	–	–	−0.7818
6	–	–	–	−0.3905
7	–	–	–	−0.1240
8	–	–	–	0.5038

Table 16
Deviation in percentage values of $N_n^{i_k,k}$ in PDF documents from the theoretical values with $n = 8$.

Runlength size (k)	Deviation in percentage values of $N_n^{i_k,k}$			
	$i_k = 4$	$i_k = 3$	$i_k = 2$	$i_k = 1$
1	−0.4338	−1.4257	−1.8391	2.2112
2	–	−0.0493	−0.0201	1.3845
3	–	–	−0.2570	−1.3554
4	–	–	–	−0.4219
5	–	–	–	−0.4593
6	–	–	–	−0.1172
7	–	–	–	0.0733
8	–	–	–	0.6871

Table 17
Root mean square deviation in percentage values of $N_n^{i_k,k}$ in application data from theoretical values with $n = 8$.

Runlength size (k)	RMS deviation in percentage values of $N_n^{i_k,k}$			
	$i_k = 4$	$i_k = 3$	$i_k = 2$	$i_k = 1$
1	0.808772	2.032438	1.015065	4.104208
2	–	0.423171	1.368368	1.917142
3	–	–	0.458404	2.572799
4	–	–	–	2.929371
5	–	–	–	1.661834
6	–	–	–	0.834875
7	–	–	–	0.401657
8	–	–	–	0.41722

and $k = 1$ and 2). In Fig. 9, we have also plotted the variation of experimentally obtained values of $N_n^{i_k,k}$ in plain text files for $n = 2048$ and $k = 1$ and 2.

5. Conclusion

We have explored, in this paper, the problem of run distribution in binary strings, and have derived an expression for $N_n^{i_k,k}$, the number of binary strings which contain exactly i_k runs of 1's of length k in all possible binary strings of length n, $1 \leq k \leq n$. The concept of generating function has been used in deriving the above expression. Our experimental results show that, for $n = 8$, the experimentally observed distributions for most file formats agree within ±5% of the

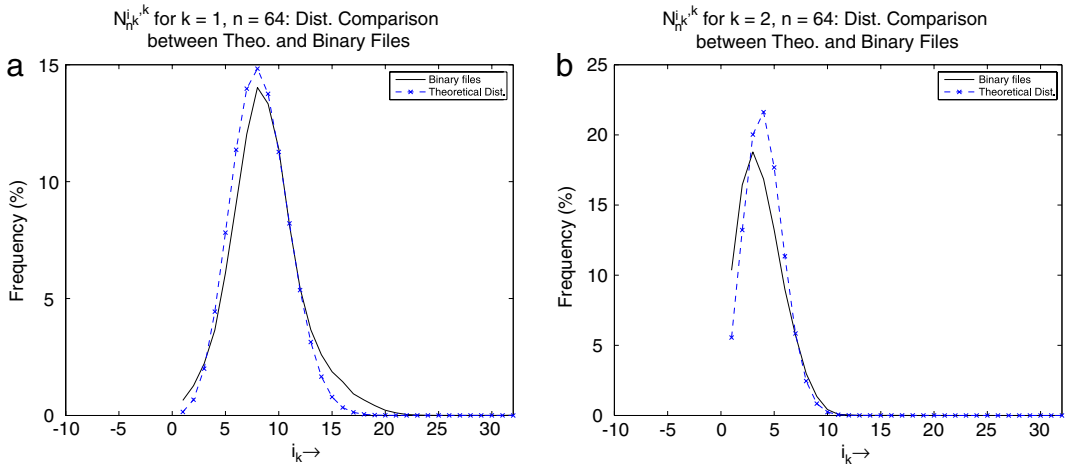

Fig. 2. Distribution of $N_n^{i_k,k}$ in binary files for $k = 1$ and 2 with $n = 64$.

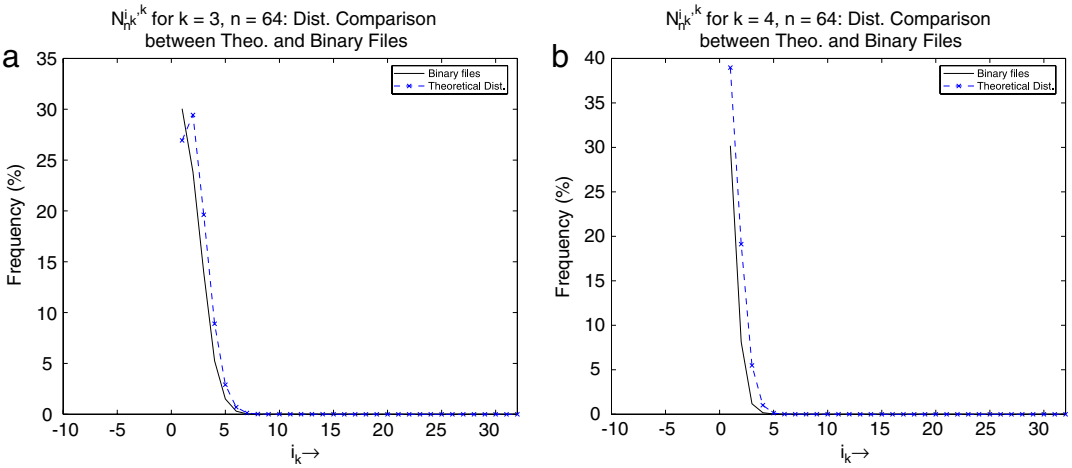

Fig. 3. Distribution of $N_n^{i_k,k}$ in binary files for $k = 3$ and 4 with $n = 64$.

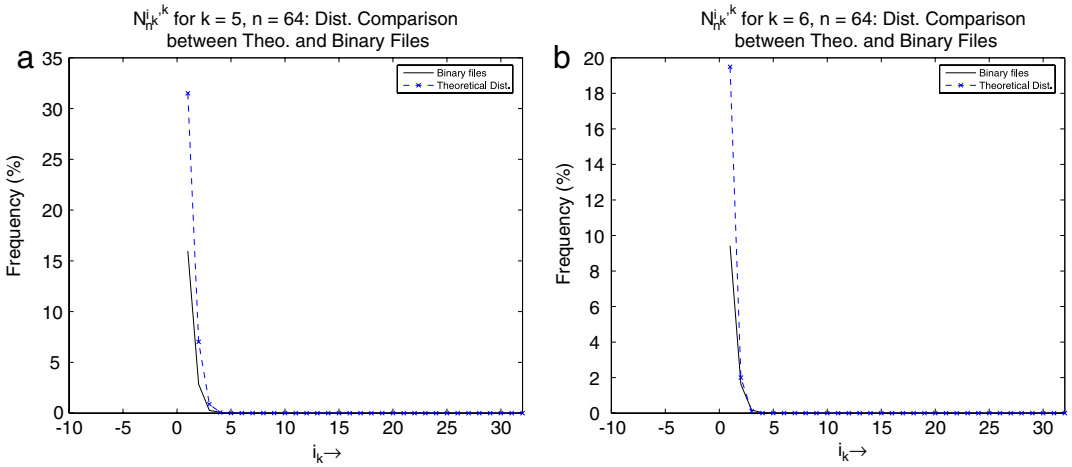

Fig. 4. Distribution of $N_n^{i_k,k}$ in binary files for $k = 5$ and 6 with $n = 64$.

theoretically obtained values and the RMS deviations of the experimentally obtained values of $N_n^{i_k,k}$ (for all i_k runs of length k, $1 \leq k \leq n$) from the theoretically derived ones, across all the file formats studied in this paper, are less

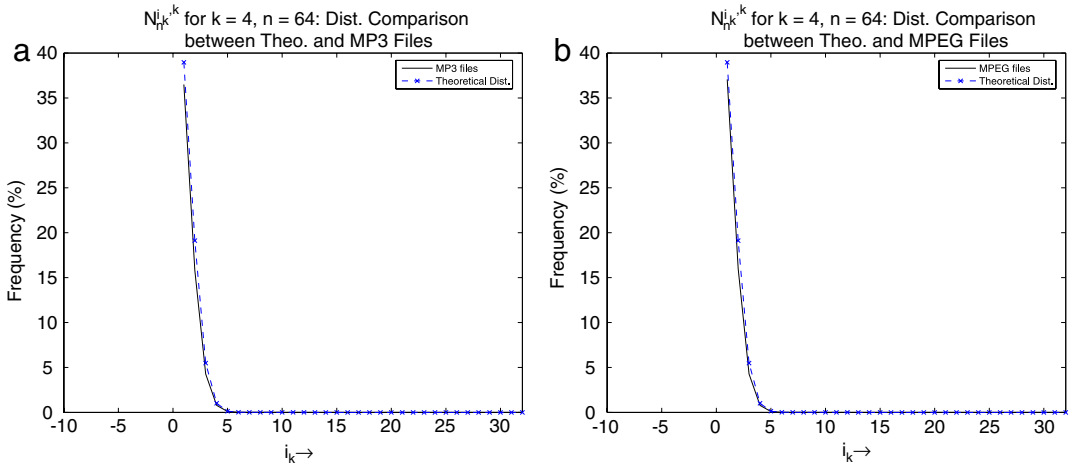

Fig. 5. Distribution of $N_n^{i_k,k}$ in MP3 and MPEG video files for $k = 4$ with $n = 64$.

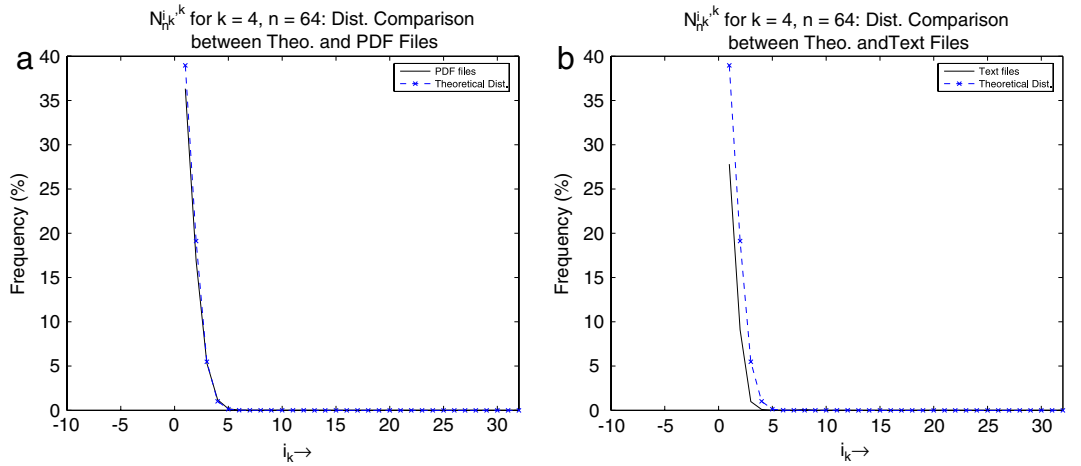

Fig. 6. Distribution of $N_n^{i_k,k}$ in PDF and text files for $k = 4$ with $n = 64$.

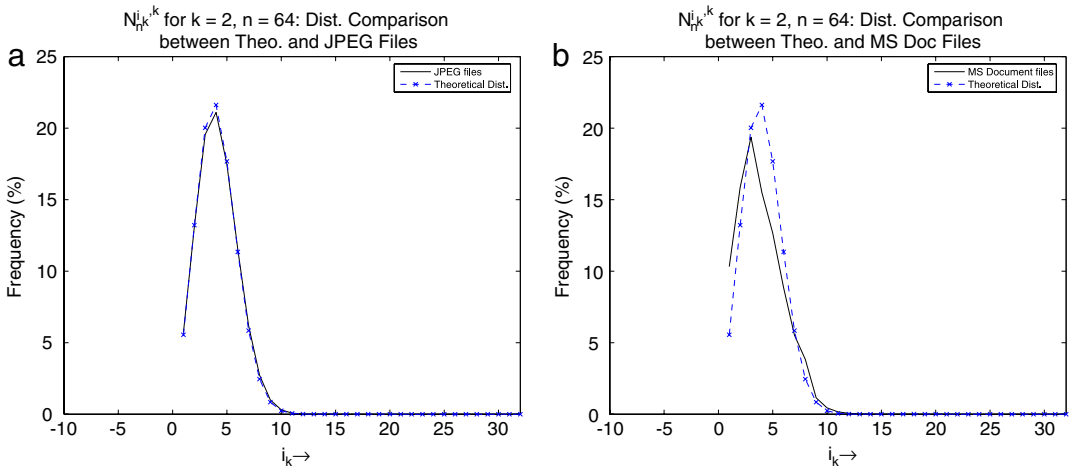

Fig. 7. Distribution of $N_n^{i_k,k}$ in JPEG and MS document files for $k = 2$ with $n = 64$.

than 5%. These results may be useful in measuring the benefit of run-length encoding in data compression, reduction in computation time in computer arithmetic with redundant binary number system, design and performance analysis of

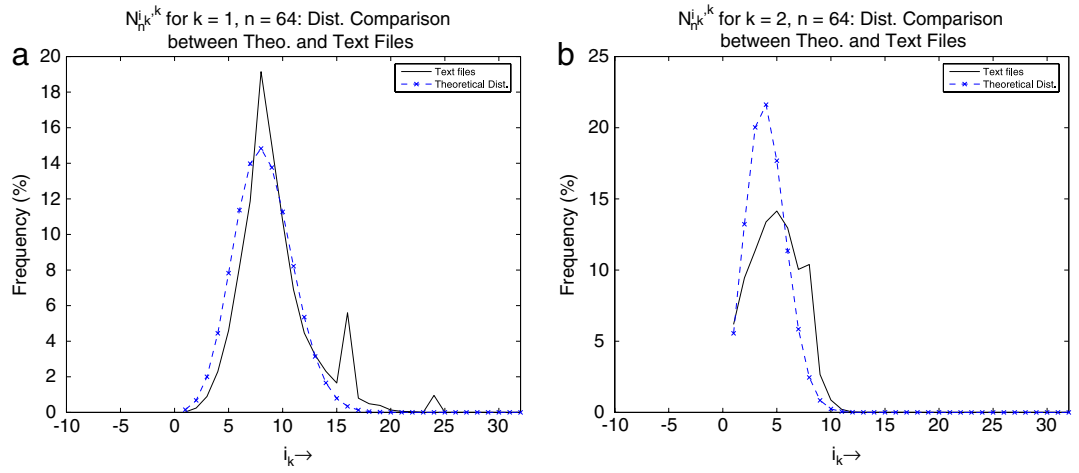

Fig. 8. Distribution of $N_n^{i_k,k}$ in text files for $k = 1$ and 2, $n = 64$.

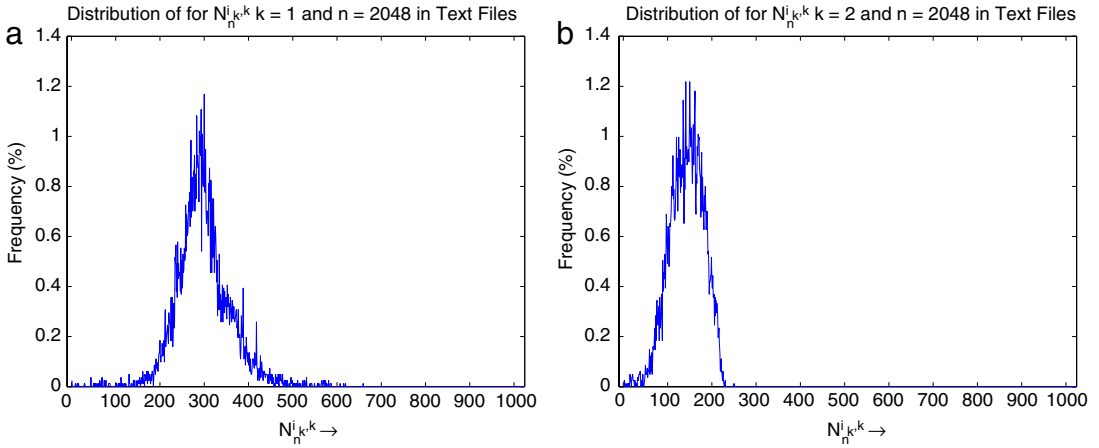

Fig. 9. Distribution of $N_n^{i_k,k}$ in text files for $n = 2048$, $k = 1$ and 2.

bus coding schemes in VLSI chips for reducing crosstalk and design of energy efficient communication schemes, among others.

References

[1] J.E. Zehavi, J.K. Wolf, On runlength codes, IEEE Trans. Inform. Theory 34 (1988) 45–54.
[2] S.T. Klein, T.C. Serebro, D. Shapira, Modeling delta encoding of compressed files, in: Data Compression Conference, 2006.
[3] J.C. Mogul, F. Douglis, A. Feldmann, B. Krishnamurthy, Potential benefits of delta encoding and data compression for HTTP, in: Proc. ACM SIGCOMM Conf. on Applications, Technologies, Architectures and Protocols for Computer Communication, 1997, pp. 181–194.
[4] C.H. Messom, G.S. Gupta, S. Demidenko, Hough transform run length encoding for real-time image processing, in: Proc. of Instrumentation and Measurement Technology Conference, IMTC, vol. 3, issue 16–19, May 2005, pp. 2198–2202.
[5] A. Nagasaka, T. Miyatake, Real-time scene identification using run-length encoding of video feature sequences, J. Robot. Mechatron. 11 (2) (1999) 98–103.
[6] N. Tagaki, H. Yassura, S. Yajima, High speed VLSI multiplication algorithm with a redundant binary addition tree, IEEE Trans. Comput. C-34 (1985) 789–796.
[7] M. De, B.P. Sinha, Fast parallel multiplication using redundant quarternary number system, Parallel Process. Lett. 7 (1) (1997) 13–23.
[8] K. Sinha, Location and communication issues in mobile networks, Ph.D. Dissertation, Dept. of Computer Science and Engineering, Jadavpur University, Calcutta, India, 2007.
[9] K. Sinha, An energy efficient communication scheme for applications based on low power wireless networks, in: Proc. 6th IEEE Consumer Communications and Networking Conference, CCNC, Las Vegas, USA, Jan. 2009, pp. 1–5.
[10] K. Sinha, B.P. Sinha, An energy efficient communication scheme for distributed computing applications in wireless sensor networks, in: Proc. Intl. Conf. on Distributed Computing and Internet Technologies, ICDCIT, in: LNCS, 2008, pp. 139–144.
[11] A. Vittal, M.M.- Sadowska, Crosstalk reduction for VLSI, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 16 (3) (1997) 290–298.
[12] Y. Shin, K. Choi, Y.-H. Chang, Narrow bus encoding for low-power DSP systems, IEEE Trans. Very Large Scale Integr. Syst. 9 (5) (2001) 656–660.
[13] M.R. Stan, W.P. Burleson, Bus-invert coding for low power I/O, IEEE Trans. Very Large Scale Integr. Syst. 3 (1995) 49–58.
[14] S. Ramprasad, N.R. Shanbhag, I.N. Hajj, A coding framework for low-power address and data buses, IEEE Trans. Very Large Scale Integr. Syst. 7 (1999) 212–221.

[15] L. Benini, G.D. Micheli, E. Macii, D. Sciuto, C. Silvano, Asymptotic zero-transition activity encoding for address busses in low-power microprocessor-based systems, in: Proc. Great Lakes Symp. VLSI, Mar. 1997, pp. 77–82.
[16] C.L. Su, C.Y. Tsui, A.M. Despain, Low power architecture design and compilation technique for high-performance processors, in: Proc. IEEE COMPCON, Feb. 1994, pp. 209–214.
[17] Y. Zhang, J. Lach, K. Skandron, M.R. Stan, Odd/even bus invert with two-phase transfer for buses with coupling, in: Proc. ISLPED, USA, Aug. 2002, pp. 12–14.
[18] Z. Khan, T. Arslan, A.T. Erdogan, Low power system on chip bus encoding scheme with crosstalk noise reduction capability, IEE Proc., Comput. Digit. Tech. 153 (2) (2006) 101–108.
[19] W.-W. Hsieh, P.-Y. Chen, C.-Y. Wang, T.-T. Hwang, A bus-encoding scheme for crosstalk elimination in hgih-performance processor design, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 26 (12) (2007) 2222–2227.
[20] W.G. Cochran, An extension of Gold's method for examining the apparent persistence of one type of weather, Q.J.R. Meteorol. Soc. 64 (1938) 631–634.
[21] A.M. Mood, The distribution theory of runs, Ann. Math. Stat. 11 (1940) 367–392.
[22] J. Wishart, H.O. Hirshfeld, On a test whether two samples are from the same population, J. Lond. Math. Soc. 11 (1936) 227–235.
[23] J. Wolfowitz, On the theory of runs with some applications to quality control, Ann. Math. Stat. 14 (1943) 280–288.
[24] J.C. Fu, M.V. Koutras, Distribution theory of runs: A Markov chain approach, J. Amer. Statist. Assoc. 89 (427) (1994) 1050–1058.
[25] F.S. Makri, A.N. Philippou, Z.M. Psillakis, Shortest and longest length of success runs in binary sequences, J. Statist. Plann. Inference 137 (2007) 2226–2239.
[26] M. Muselli, Simple expressions for success runs distributions in Bernoulli trials, Statist. Probab. Lett. 31 (1996) 121–128.
[27] V.T. Stefanov, On run statistics for binary trials, J. Statist. Plann. Inference 87 (2000) 177–185.
[28] N. Balakrishnan, M.V. Koutras, Runs and Scan with Applications, John Wiley, New York, 2002.
[29] W. Feller, An Introduction to Probability Theory and its Applications, vol. 1, 3rd ed., John Wiley, New York, 1968.
[30] K.D. Ling, On binomial distributions of order *k*, Statist. Probab. Lett. 6 (1988) 247–250.
[31] J.D. Gibbons, Nonparametric Statistical Inference, McGraw-Hill, New York, 1971.
[32] E. Musoll, T. Lang, J. Cortadella, Exploiting the locality of memory references to reduce the address bus energy, in: Proc. Intl. Symp. Low Power Electronics Design, 1997, pp. 202–207.
[33] D.I.A. Cohen, Basic Techniques of Combinatorial Theory, John Wiley & Sons Inc., New York, 1979.
[34] C.L. Liu, Introduction to Combinatorial Mathematics, McGraw-Hill, New York, 1968.
[35] H.S. Wilf, Generating Functionolgy, 2nd ed., Academic Press Inc., 1994.